Pliny Deployment Guide

Introduction

This document details the deployment procedures for the Pliny application, a visualization tool for analyzing trends in Wikipedia metadata. The deployment environment consists of a single Google Cloud Platform (GCP) virtual machine hosting both backend and frontend components. Additionally, you'll need a local machine with the Pliny repo installed to build and test both components.

System Architecture

Pliny consists of three primary components:

1. Data Pipeline (Python): Processes Wikipedia data from APIs or dumps
2. Backend (Go): Serves processed data via REST API
3. Frontend (TypeScript/React): Web application for data visualization

Note that only the backend and frontend are deployed. The data pipeline is run as a Python script to ingest data into a database that's then served by the other components.

Deployment Environment

Infrastructure

- **Hosting**: Google Cloud Platform
- **VM Instance**: `pliny-backend` (Debian)
- **Web Server**: NGINX (serving frontend)
- **Process Manager**: tmux (for backend processes)

Network Configuration

- Backend port: 8080
- Frontend served on standard HTTPS port via NGINX

Deployment Procedures

Backend Deployment

The backend is run in a seperate shell with `tmux`, and can simply be run with the standard go command. Steps:

1. **SSH Access**

   ```
   # Access the VM via the GCP console's SSH button
   ```

2. **Access Running Process**

   ```
   tmux ls                    # List running tmux sessions
   tmux a                     # Attach to the running session
   ```

3. **Update and Restart Backend**

```
# While attached to tmux session
CTRL-C                      # Stop the current process
git pull                    # Pull latest changes

# Ensure the router is configured for external access in main.go:
# router.Run("0.0.0.0:8080")

go run *.go                 # Start the backend
```

4. **Detach from tmux Session**

```
CTRL+B, then D              # Detach from tmux session
```

Frontend Deployment

To deploy the frontend, you'll first need a static build that is generated from the React files. Since the backend server is very lightweight and doesn't have `npm` isntalled, you'll have to generate these static files on your local machine. Steps:

1. **Local Build**

```
# In your local /pliny/frontend directory
npm run build               # Generate production build
zip build.zip -r build      # Compress build directory
```

2. **Upload to Server**

   - Use the "Upload File" feature in GCP SSH interface
   - Upload the `build.zip` file to the home directory (~)
   - Alternatively, you can use `scp` with a key you generate on the GCP terminal. However, this is more effort than just using the web interface.

3. **Deploy to Web Server**

```
# On the remote server
unzip build.zip -d frontend_build      # Extract files
sudo cp -r ./frontend_build/build/* /var/www/html/  # Copy to NGINX directory
sudo systemctl restart nginx           # Restart web server
```

4. **Cleanup**

```
rm -rf frontend_build build.zip        # Remove temporary files
```

Troubleshooting

Backend Issues

- Check tmux session is running: `tmux ls`

- Verify backend is listening on correct port: `netstat -tulpn | grep 8080`
- Review logs in the tmux session: `tmux a`

Frontend Issues

- Verify NGINX configuration: `sudo nginx -t`
- Check NGINX status: `sudo systemctl status nginx`
- Review NGINX logs: `sudo tail -f /var/log/nginx/error.log`

Data Pipeline Deployment

The data pipeline component is not part of the regular deployment process as it is run on-demand to update the dataset. Refer to the data-pipeline/README.md for specific instructions on running the data processing jobs.

Environment Variables

A `.env` file is required for BigQuery access. Ensure this file is properly configured on the deployment environment with the appropriate credentials for database access.

If you create a new VM to run the Pliny backend on, you'll have to create the `.env` file yourself witht the correct credentials.

Security Considerations

- The GCP VM should have appropriate firewall rules to restrict access
- HTTPS should be configured for production environments. Currently this is done through NGINX with Let's Encrypt.
- BigQuery credentials should be kept secure and not committed to version control